I claim:

1. A data-sharing method wherein a non-cooperative DBMS of a primary computer system participates in unaware applications and has a cache, respective lock structures, database log files and database data files responsive to data requests generated by the unaware applications, said method comprising the steps of:

(a) nonintrusively monitoring data written to said database log files and said database data files and communicating information as to data written to said files to a secondary DBMS running on a potentially different computer and having a secondary cache and secondary lock requests and responsive to data requests by other unaware applications; and

(b) processing data in said secondary DBMS between said other unaware applications and with said secondary cache and said secondary lock requests while reading data from said non-cooperative DBMS data files without interrupting update or retrieval activities of said non-cooperative DBMS and while isolating said non-cooperative DBMS from said other applications, thereby enabling said other unaware applications to access the data maintained by said non-cooperative DBMS.

2. The method defined in claim 1, further comprising the step of intercepting data written by said primary computer system to said non-cooperative DBMS and parsing the intercepted data nonintrusively with respect to said primary computer system with a listener and utilizing the parsed intercepted data to establish said secondary cache and secondary lock requests shielding said other unaware applications from inconsistent data of said primary computer system.

3. The method defined in claim 1wherein said secondary DBMS operates with items of interest having a structure consisting of a part defining an item type distinguishing between parts of a data base, a part defining an identity of the item in the database, a "dirty" part describing parts of an item not previously transferred to storage, a part describing a previous transaction involving the item to permit updating of that transaction, a part describing a locking transaction, a part facilitating application of an optimization algorithm, a list of pending reads identifying processes which have shown interest in

the item, a part representing a before image constituting a pointer to data represented by the item before the transaction, a part representing an after image of data subsequent to the transaction and a part representing a transaction initiated by a respective one of said other unaware applications.

4. The method defined in claim 3 wherein each transaction with an item of interest is effected in said listener by the steps of:

(i) check whether the item of interest is a first item of the transaction;

(ii) if the item of interest is the first item of the transaction, process the item of interest by exclusively locking the transaction identification of the item of interest and creating a transaction entry therefore;

(iii) if the item of interest is not the first item of the transaction or after the creation of the transaction entry, check whether the item of interest is already in cache;

(iv) if the item of interest is not already in cache, create a cache entry therefore;

(v) if the item of interest is already in cache, check whether the item of interest belongs to a previous transaction;

(vi) if the item of interest is already in cache but does not belong to a previous transaction and following step (iv), concatenate the cache entry with a transaction context and create the before image for the item of interest; and

(vii) following step (vi) and, where the item of interest following step (v) belongs to a previous transaction, updating the cache entry to contain a new after image and "dirty" part.

5. The method defined in claim 3, further comprising checking an item of interest read by said listener in the past by the steps of:

(I) checking whether the item of interest is locked by a transaction;

(II) if the item of interest is not locked by a transaction, verifying if the previous transaction for the item of interest is the same as the previous transaction for a prior reading by comparing the previous transaction part of the item of interest with a corresponding entry of the previous transaction at the prior reading; and

(III) validating the item of interest when the previous transaction part is the same as the corresponding entry of the previous transaction at the prior reading.

6. The method defined in claim 3, further comprising initiating at time intervals a postlistener sequence in said listener which comprises the steps of:

(A) scanning an item of interest cache entry;

(B) selecting entries of items of interest having NULL locking transaction parts;

(C) for each item of interest having a NULL locking transaction part, checking whether the item of interest entry has a "dirty" part;

(D) for each item of interest found to have a "dirty" part in step (C), reading corresponding data from storage and updating the "dirty" part data; and

(E) following step (A) in the case of a cache entry of an item of interest having a not NULL locking transaction part, following step (C) for each cache entry having no "dirty" part and following step (D) for each cache entry having an updated "dirty" part, returning to step (A) to scan a next item of interest cache entry until all item of interest cache entries are scanned.

7. The method defined in claim 3 wherein update operations initiated in said secondary computer system are delegated to the non-cooperative DBMS by the steps of:

transmitting from said secondary DBMS to said non-cooperative DBMS of said primary computer system an update instruction based upon a read transaction of one of said other unaware applications;

checking whether the update of the non-cooperative DBMS of the primary computer system has been completed;

thereafter locking the transaction initiated by said one of said other unaware applications; and

creating cache entries including after images for all items of interest affected by the update.

8. The method defined in claim 2, further comprising the step of updating, with said secondary DBMS in response to one of said other unaware applications, the non-cooperative DBMS at least in part by delegating update operations and subsequent retrieval operations directly or indirectly to said non-cooperative DBMS.

9. The method defined in claim 1 wherein the cache associated with said secondary DBMS is provided with a storage capacity sufficient to hold the entire contents of the data of at least one of said DBMSs.

10. The method defined in claim 1 wherein said primary computer system has a controller, said method further comprising the step of speeding a response time of said non-cooperative DBMS and reducing an I/O load on said controller by intercepting and eliminating the physical write operations directed to a database of the non-cooperative DBMS, and utilizing said controller to intercept log writes to automatically trigger writes to a disk of the non-cooperative DBMS, and for directly responding to read requests relying on data cached in memory of said non-cooperative DBMS.

11. The method defined in claim 1 wherein at least one of said DBMSs is provided with a time function keeping a transaction time for an entire duration of each transaction and creating an appearance of a transaction occurring at a single point in time, thereby supporting true repeatable read and serializable transactions.

12. The method defined in claim 11 wherein, for each transaction requiring a ''repeatable read'' or ''serializable'' isolation level, a transaction snapshot entry is created which includes a transaction identifier identifying the transaction, a log position identifier containing a last position in the log that has been read at the instant the transaction is initiated, a time stamp serving for date-related functions required by the logic for the transaction and a process identifier pointing to a process which issued the transaction.

13. A computer system comprised of a secondary DBMS, a secondary cache and secondary lock structures and connectable for data sharing with a non-cooperative DBMS of a primary computer which participates in unaware applications and has a cache, respective lock structures, database log files and database data files responsive to data requests generated by the unaware applications, said computer system having a listener connected to said non-cooperative DBMS for:

(a) nonintrusively monitoring data written to said database log files and communicating information as to data

written to said files to said secondary DBMS of said computer system, said computer system being responsive to data requests by other unaware applications; and

(b) processing data in said secondary DBMS between said other unaware applications and with said secondary cache and said secondary lock requests while reading data from said non-cooperative DBMS with said secondary DBMS without interrupting update or retrieval activities of said non-cooperative DBMS and while isolating said non-cooperative DBMS from said other applications, thereby enabling said other unaware applications to access the data maintained by the non-cooperative DBMS.

14. The computer system defined in claim 13 wherein said listener, said secondary DBMS and said secondary cache are provided in a single hardware unit separately from a computer on which said other applications are run.

15. The computer system defined in claim 13 wherein said listener, said secondary DBMS and said secondary cache are provided in a computer in which said other applications are run.

16. The computer system defined in claim 13 wherein said listener is connected to intercepting data written by said primary computer to said non-cooperative DBMS and is programmed to parse the intercepted data nonintrusively with respect to said primary computer and utilize the parsed intercepted data to establish said secondary cache and secondary lock requests shielding unaware applications executed in said computer system from inconsistent data of said primary computer.

17. The computer system defined in claim 13 wherein said secondary cache is provided in a memory with a storage capacity corresponding to the entire database of the non-cooperative DBMS.

18. The computer system defined in claim 14 wherein said listener and said secondary DBMS are combined with a storage controller responsive to SQL select and other data manipulation read commands retrieving data of said non-cooperative DBMS.

19. The computer system defined in claim 18 wherein at least one of said computers has a multiplicity of dedicated CPUs for executing the SQL select and other data manipulation read commands in parallel on different parts of the secondary cache.

20. A method of supporting true repeatable read and serializable transactions which comprises the steps of:

(a) during a repeatable read and serializable transaction in a database management system, storing a transaction time for an entire duration of the transaction; and

(b) automatically based on the transaction time stored in step (a) further processing said transaction to create an appearance that said transaction occurred at a single point in time.

ge-